REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 02-07-2002	**2. REPORT DATE** Performance/Technical	**3. DATES COVERED** *(From - To)* May 1, 2001-April 30, 2002

4. TITLE AND SUBTITLE Building Interactive Digital Libraries of Formal Algorithmic Knowledge.	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER N00014-01-1-0765
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Robert L. Constable	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Cornell University, Computer Science Dept. 4130 Upson Hall Ithaca, NY 14853	**8. PERFORMING ORGANIZATION REPORT NUMBER** 39544
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research, Navy, Dept. of Defense 800 N. Quincy St. Arlington, VA 22217-5660	**10. SPONSOR/MONITOR'S ACRONYM(S)** ONR
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; distribution is Unlimited

20020708 096

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This report summarizes the conceptual basis and the first prototype of an Interactive Digital Library of Formal Algorithmic Knowledge. The key purpose of the prototype library is to demonstrate that it is possible to organize formal knowledge that is necessary for the development of reliable hardware and software and to provide software tools for using it in a variety of ways.

15. SUBJECT TERMS
Digital libraries, algorithmic knowledge, theorem proving.

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Juanita Heyerman
U	U	U	UU	7	**19b. TELEPONE NUMBER** *(Include area code)* 607/255-5331

Annual Report

RE: Contract Number ONR N00014-01-1-0765,
"Building Interactive Digital Libraries of Formal Algorithmic Knowledge."

Contract Number: ONR N00014-01-1-0765

Reporting Period: May 1, 2001 - April 30, 2002

Submission Date: July 2, 2002

Prepared by:
Robert L. Constable
Cornell University
4130 Upson Hall
Ithaca, NY 14853

Building Interactive Digital Libraries of Formal Algorithmic Knowledge

Cornell University
Annual Report, May 1, 2001 - April 30, 2002

Contract Number: N00014-01-1-0765

Background:

Achievements of mathematicians, logicians, and computer scientists over the past fifty years have created the practical means to formalize vast amounts of mathematical knowledge. A large collection of formal material has arisen from applications; included therein is a large number of general mathematical results needed to support those applications. The volume of material increases daily.

This formal material presents extraordinary opportunities and challenges. The opportunity is to organize the material so that it is more widely usable and shared. It is valuable in creating more reliable hardware and software and in expanding the capacity of many formal tools needed to protect the software infrastructure of the nation.

As an artifact in itself, the collection of formal material has exceptional properties. It is digital, logically organized, and highly structured. It codes vast amounts of mathematical knowledge, especially algorithmic knowledge and represents the highest standards of correctness and accuracy currently imaginable. It captures the precise thinking of a large number of excellent scientists who have spent hundreds of person years in creating this as yet unorganized collection with limited accessibility.

Our long-term goal is to organize this formal knowledge and to provide software tools for using it in a variety of ways. Another long-term goal is to enable a worldwide user community to contribute new formal material and to contribute original articles that incorporate this material in aid of ordinary scientific and educational discourse.

Conceptual Basis:

On a conceptual basis, we have progressed in two major areas: (1) separation of functions and responsibilities and (2) design of the library functionality.

The first area reflects the conflict between supporting a broad collection of logics and content, and ensuring the "correctness" of the library. The basic library architecture does not specify proof structure, but instead stipulates how clients may introduce certification methods. Similarly, there is no basic notion of mathematical definition, a very generic term structure, and no large organization structure between formal mathematical objects. Instead, all these structures will be represented explicitly by library objects. Our basic library design is open in two ways: a library process should be accessible by various clients and other parties should be able to build their own implementations of the library.

The collection of concepts on which the library design is based has solidified significantly as a result of the discussions and extensive technical notes among the group, as well as with outside council (William Arms, Alan Demers, Johannes Gehrke, Paul Ginsparg, John Kleinberg, and Carl Lagoze). For the purpose of design we have given explicit meanings to concepts of special technical significance, such as: Certificate; Native Language; Abstract Identifier and External Name; Term; Library Object and Object Reference; Client and Session; Proof, Inference Step, Assertion, Justification, Lemma, Inference Engine and Proof Sentinel. Definitions of each of these are given in the FDL technical report listed below.

Prototype Formal Digital Library:

We have implemented a prototype implementation of the Formal Digital Library that allows people to read, organize, search, and annotate formal content and to produce new formal content in the form of definitions, theorems, theories, proof methods, and articles about topics in computational mathematics. The library provides the mechanisms that account for the integrity of the formal content, using certificates to attest that certain actions (like calling proof engines) were taken to validate the contents of library objects.

Our prototype is organized as a persistent object store that preserves previous versions when the contents of an object are modified. This ensures durability of information and replayability of proofs that were accepted by the library. A version control mechanism makes it possible to recover previous versions of an object. This protects user data from being corrupted or destroyed erroneously and enables a user to keep several versions of the same object, while developing the contents of a formal algorithmic theory.

Proof Engines:

Proof engines provide mechanisms for interactive, tactical, and fully automated reasoning in a specific formal language and generate justifications for the formal content stored in the Library. In the first project year we have:

– Developed an API module that connects the Library to the Nuprl proof engine, which supports interactive and tactic-based reasoning in computational type theory within a sequent calculus framework. We have imported all the basic Nuprl rules, tactics, and theories on which they depend into the Library in order to be able to control the refiner's behavior through code objects in the library. We have also added mechanisms for tactics to run asynchronously on incomplete proof goals, which makes it possible to run expensive tactics in the background while users continue to work on other proof goals.

– Developed an API module that connects the Library to the MetaPRL system, which supports interactive and automated reasoning in multiple logics. The module convert each system's data to a common MathBus interchange format, which makes it possible to send commands and their arguments to MetaPRL from the Library and to send the resulting proofs from MetaPRL to the Library.

– Developed and implemented JProver, a completely automated theorem prover for intuitionistic and classical first-order logic that is capable of producing sequent-style proofs, in which each individual proof step is justified by a basic inference rule. To connect JProver to the library, we have developed an API module that converts library contents into the language of JProver and vice versa.

– Implemented an API module for connecting the Library to the PVS system, which provides mechanized support for formal specification and verification based on a classical, typed higher-order logic. The module converts library contents into PVS notation, sends sequents and PVS commands to the PVS proof engine, and builds library terms from the results.

Linking Libraries and Migrating Formal Content:

In order to use the Library as a common repository, it has to be linked to the libraries of external proof systems, whose contents are then migrated into the Library. This involves converting formal expressions into the Library format and constructing inference trees, proofs, and certificates from the proofs build with the respective systems. In the past year we have developed migration packages for Nuprl-5, Nuprl-4, MetaPRL, and PVS and migrated the corresponding library contents into the Library.

– The term structure of Nuprl-5 system is similar to that of the Library. Migrating Nuprl-5 contents into the library thus essentially meant importing the theories into the Library while rebuilding certificates.

– Users of the Nuprl-4 system have developed a substantial amount of formalized mathematical knowledge. To integrate this knowledge into the Library, the migration package converts Nuprl-4 data into the more general format and replays the formal theories by checking formal proofs with the Nuprl-5 proof engine, which builds the appropriate certificates. A key difficulty of the migration was ascertaining dependencies and specifying the state of the proof engine, which was done in Nuprl-4 to control proof caching.

– MetaPRL's logics (i.e. CZF, ITT) are organized in separate theory modules. Library contents are migrated from MetaPRL into the Library by converting them into the MathBus interchange format. Then the proofs are checked by calling the MetaPRL proof engine, which builds the appropriate certificates.

– Users of PVS have developed large repositories of formal knowledge about the formal specification and verification of software. To migrate PVS theories into the Library, the Library sends commands to the PVS system that make it display all its theories and proofs. A ``PVS parser" then builds Library terms from ASCII representations of PVS expressions. Proofs are re-executed with the connected PVS proof engine in order to build the necessary certificates.

In addition to these proof systems, we have also developed an XML Parser that enables us to import XML data into the Library. This is particularly important for application projects in software reliability.

Publishing and Reading:

In order to use the Library as a common repository, it has to be linked to the libraries of external proof systems, whose contents are then migrated into the Library. This involves converting formal expressions into the Library format and constructing inference trees, proofs, and certificates from the proofs build with the respective systems. In the past year we have developed migration packages for Nuprl-5, Nuprl-4, MetaPRL, and PVS and migrated the corresponding library contents into the Library.

One of the key services that a library must provide is an interface that makes formal algorithmic knowledge accessible to users. We have developed two publication mechanisms that enable external users to access the contents of the logical library in a simple way.

– We have developed mechanisms for automatically producing LaTeX documents from the contents of a formal theory. These mechanisms enable the producer of a formal theory to build structured documents inside the system. These documents consist informal ``articles" that contain links to formal content such as definitions, theorems, and proofs. Display forms make it possible to describe the presentation of these documents on the screen and in a printed document.

– A web interface processes library information for publication on the web. It analyzes the links between formal and informal objects and creates web booklets that present the material on several levels of detail. On the top level, a user will look at a formatted technical report. Clicking on the formal (top-level) text in this document reveals the further details at all levels of precision – from the rough sketch of a theory or proof down to the level of the underlying logic. We expect that the ability to unveil formal details on demand will have a significant educational value for teaching and understanding mathematical and algorithmic concepts.

Design Documents:

– FDL project web page www.nuprl.org/html/Digital_Libraries.html

– Nuprl Mathematics Library www.nuprl.org/Nuprl4.2/Libraries

Publications:

– JProver: Integrating Connection-based Theorem Proving into Interactive Proof Assistants. (Stephan Schmitt, Lori Lorigo, Christoph Kreitz, Aleksey Nogin) In R. Gore, A. Leitsch, T. Nipkow, eds., International Joint Conference on Automated Reasoning (IJCAR 2001), LNAI 2083, pp. 421–426, Springer Verlag, 2001.

– Computational Complexity and Induction for Partial Computable Functions in Type Theory. (Robert Constable, Karl Crary) In W. Sieg, R. Sommer, C. Talcott, eds., Reflections on the Foundations of Mathematics: Essays in Honor of Solomon Feferman, pp. 166–183, Association for Symbolic Logic, 2001.

– Proving Hybrid Protocols Correct. (Mark Bickford, Christoph Kreitz, Robbert van Renesse, Xiaoming Liu.) In R. Boulton & P. Jackson, eds., 14th International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2001), LNCS 2152, pp. 105–120, Springer Verlag, 2001. TPHOLS

– Designing Reliable, High-Performance Networks with the Nuprl Logical Programming Environment. (Christoph Kreitz) 2002 AAAI Spring Symposium on Logic-Based Program Synthesis, Stanford University. Technical Report SS-02-05, AAAI Press, pp. 55–56, 2002.

– Naive Computational Type Theory. (Robert Constable) In H. Schwichtenberg & R. Steinbrueggen, eds., Proof and System-Reliability, NATO Science Series III, pp. 213–260, Kluwer Academic Publishers, 2002.

– FDL: A Prototype Formal Digital Library. (Stuart Allen, Mark Bickford, Robert Constable, Richard Eaton, Christoph Kreitz, Lori Lorigo) Technical Report, Cornell University, Ithaca, NY, May 2002.